Application Layer Multicast Algorithm

Sergio Machado and Javier Ozón
Universitat Politècnica de Catalunya
Castelldefels

Abstract

This paper presents a multicast algorithm, called MSM-s, for point-to-multipoint transmissions. The algorithm, which has complexity $O(n^2)$ in respect of the number n of nodes, is easy to implement and can actually be applied in other point-to-multipoint systems such as distributed computing. We analyze the algorithm and we provide some upper and lower bounds for the multicast time delay.

1 Introduction

Multicast is a point-to-multipoint means of transmitting data in which multiple nodes can receive the same information from one single source. The applications of multicast include video conferencing, multiplayer networking games and corporate communications. The lack of deployment of IP Multicast has led to considerable interest in alternative approaches at the application layer, using peer-to-peer architectures[7]. In an application layer multicast approach, also called overlay multicast, participating peers organize themselves into an overlay topology for data delivery. In this topology each edge corresponds to an unicast path between two end-systems or peers (also called nodes) in the underlying IP network. All multicast-related functionality is implemented by peers instead of routers, with the goal of depicting an efficient overlay network for multicast data transmission.

In this work we present an algorithm suitable for peer-to-peer multicast transmissions, although the high degree of abstraction of its definition makes it also suitable for its implementation in other layers and in general message-passing systems. The main contribution of this proposal is that the operation of our algorithm is simple, with a complexity of $O(n^2)$, where n is the number of peers, and thus it may adapt dynamically to the characteristics of the source and the network in order to improve the multicast time delay. Algorithm execution may be computed by a single group member, usually the one which multicasts the message, or by all the members after the complete network status has been broadcasted.

2 Single Message Multicast Algorithm

Bar Noi *et al.* introduced in [1] the *Message Passing System Model* $\mathcal{MPS}$ which characterizes systems that use packet switching techniques at the communication network. In this work, we extend the Bar Noi model to $E\mathcal{MPS}(n, \boldsymbol{\lambda}, \boldsymbol{\mu})$, which consists of a set of n full-duplex nodes $\{p_0, \ldots, p_{n-1}\}$ such that each node can simultaneously send and receive a message. The term *message* refers to any atomic piece of data sent by one node to another using the protocols of the underlying layers. For each node p we define the *transmission time* μ_p as the time that requires p to transmit a single message. Moreover, for each pair of nodes p and q in a message-passing system we define the *communication latency* λ_{pq} between p and q as follows. If at time t node p starts to send a single message to node q, then node p sends the message during the time interval $[t, t + \mu_p]$, and node q receives the message during the time interval $[t + \lambda_{pq} - \mu_p, t + \lambda_{pq}]$. Thus λ_{pq} is the transmission time μ_p of node p plus the propagation delay between p and q, as shown in Figure 1. We denote by $\boldsymbol{\mu}$ the vector of all μ_p's and by $\boldsymbol{\lambda}$ the matrix of all λ_{pq}'s in the network. For simplicity sake, we assume that the communication latency is constant, and we consider multicast as a broadcast problem, since we can isolate the receiving nodes of a multicast communication, form with them a complete overlay graph, and then depict a routing table through a broadcast algorithm.

Let p_0 be the source node in $E\mathcal{MPS}(n, \boldsymbol{\lambda}, \boldsymbol{\mu})$ model which has a message to multicast to the set of receiving nodes $R = \{p_1, p_2, \ldots, p_{n-1}\}$, we search for an algorithm that minimizes the *multicast time*, that is, the time at which all nodes in R have received the message. Though the result of $E\mathcal{MPS}$ is a multicast spanning tree, in Figure 2 we show that this problem is different from the well known minimum spanning tree problem.

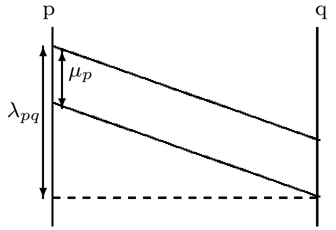
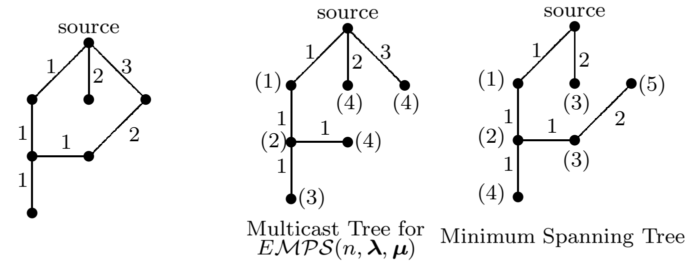


Figure 1: The Postal Model. The latency λ_{pq} is equal to transmission time μ_p plus the propagation delay between p and q.



Multicast Tree for $\mathcal{EMPS}(n, \boldsymbol{\lambda}, \boldsymbol{\mu})$

Minimum Spanning Tree

Figure 2: Example of a network where $\mathcal{EMPS}(n, \boldsymbol{\lambda}, \boldsymbol{\mu})$ model does not correspond to the MST problem. We show in parenthesis the multicast delay for each node in the case that transmission time μ is 1 for any node.

The algorithm that we propose, called SMM *Single Message Multicast*, operates as follows: at each step SMM algorithm chooses the node which has not yet received the message and has the lowest cost, that is, the unvisited node that can be reached at minimum time from the nodes which have already received the message. Once the message has been received by this node, the algorithm recalculates the arrival times of the remaining nodes, searches the next node at which the message must be forwarded, and so forth. We assume that when a sending node finishes the retransmission of the message to another node, it begins immediately with another destination node. SMM algorithm is very similar to Dijkstra's shortest path algorithm with the difference that in this case the time delay between two nodes p and q is not constant. Actually, in $\mathcal{EMPS}(n, \boldsymbol{\lambda}, \boldsymbol{\mu})$ this delay is equal to λ_{pq} plus μ_p multiplied by the number of previous retransmissions of node p.

The multicast time achieved by the algorithm SMM is minimum when $\mu_p = 0$ for all the nodes. In this case, the time delay between two nodes p

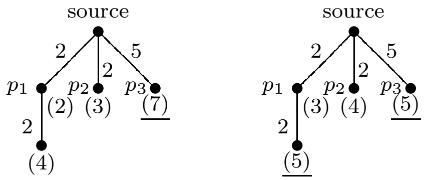


Figure 3: Example of a network where SMM is not optimal. Transmission time is 1 for all the nodes. On the left we apply SMM and on the right another multicast transmission order with a better result.

and q is always the weight λ_{pq} of the edge which joins them, and thus the SMM algorithm corresponds to the optimal algorithm Dijkstra of complexity $O(n^2)$. In a general case, however, the SMM algorithm is not always optimal. Figure 3 shows a network where SMM is not optimal.

Proposition 1 *Algorithm SMM for* $\mathcal{EMPS}(n, \boldsymbol{\lambda}, \boldsymbol{\mu})$ *has complexity* $O(n^2)$.

Proof: At each step SMM searches the node which has not yet received the message with lowest cost. As the maximum number of unvisited nodes is $n-1$ this operation requires at most $n-2$ comparisons. Moreover, the algorithm executes one step for every node which receives the message. Thus, we have $n-1$ steps and at each step we perform at maximum $n-2$ comparisons plus some basic and bounded operations resulting in a complexity of $O(n^2)$. □

2.1 Message Stream Multicast Algorithm

The SMM algorithm has been defined for the multicast of a single message. For a set of messages we can repeat indefinitely the routing table obtained with the SMM algorithm, multicasting each message independently of the others. That means that when one message finally arrives at all the nodes, the message source would proceed to multicast the next message, and so forth. The total delay multicast time of the stream would be in this case the total number of messages M multiplied by the multicast SMM delay for one single message. The main inconvenience of this solution is that the source can not send the next message until the previous one has been received by all the group members and this could slow down the rate of communication.

Next, we consider a new possibility. Before the first message has arrived at all nodes, the source could stop sending it and begin with the second message. With this restriction, the multicast time of the first message will increase, but we will begin to send before the second message. This saving of time between the sending of two consecutive messages will be progressively accumulated, and if the number of messages is large enough it will compensate the increase of the multicast time for one single message. The modified algorithm, that we call MSM-s *Message Stream Multicast*, works as SMM and applies the same multicast scheme for every message with the particularity that it stops the transmission of any message once a node has already sent it s times. Then it will begin to send the next message and so forth. Since the restriction on the number of retransmissions could isolate some nodes of the network, MSM-s should choose a minimum restriction number s to guarantee full-connectivity. As SMM, MSM-s algorithm has complexity $O(n^2)$.

In next sections we prove that under certain conditions it is possible to calculate a minimum number M_σ in such a way that if the number of messages is equal or larger than M_σ then MSM-σ is better than MSM-$(\sigma + 1)$. Moreover, when restricting the number of transmissions for each node, MSM-s has to take into account the transmission rate of the source. That is, if the source sends at most s times the first message and then, after $s \cdot \mu_r$ time units, stops the transmission of the first message to begin with the second one, we must be able to assume that the source has the second message ready to forward. Otherwise, the source would stop sending the first message before having the second one and would remain unnecessarily idle, with the consequent loss of efficiency.

2.2 Message Stream Multicast Algorithm with Time Restriction

Let p be a node which forwards the message to node q, and let $s_p(s), s_q(s) \leq s$ be the times that p and q forward the message for MSM-s, respectively. In this case the second message will be received at q with a delay of $s_p(s) \cdot \mu_p$ in respect to the first message, since the second message follows the same path but with a source delay of $s_p(s) \cdot \mu_p$, as seen in Figure 4. When the forwarding period $s_q(s) \cdot \mu_q$ of node q is higher than the forwarding period $s_p(s) \cdot \mu_p$ of node p, then successive messages may have higher delays than former messages. In this context, the second message could arrive at node

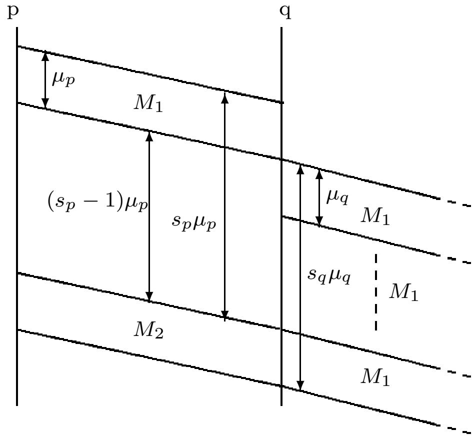


Figure 4: The limit s_q of retransmissions of peer q could be different for every peer q, depending on the transmission times of the peers.

q before it has finished forwarding the first message and then the second message would have to be buffered, with the consequent time delay. This buffering delay would be also accumulated by the third message, and so forth. Nevertheless, this situation may be avoided by limiting the time period $s_q(s)\cdot\mu_q$ at which each node forwards a message, that is, by assuring that the forwarding rate $1/(s_q(s)\cdot\mu_q)$ of any node q is higher than the rate $1/(s_p(s)\cdot\mu_p)$ of any node p which is in the path from the source to node q, including the source. Therefore, the delay of the first message will be always the same as the time delay of any other message, an issue which has great importance in Section 3. Note also that we do not want $1/(s_q(s)\cdot\mu_q) \gg 1/(s_p(s)\cdot\mu_p)$ since in this case q would stop forwarding the first message long before receiving the second one, and then the communication would lose efficiency.

3 Analysis of MSM-s

3.1 Stream Multicast Delay

Let τ_s be the multicast time delay for a single message when the number of transmissions of each node is established up to s, M the number of messages of the stream and μ_r the transmission time of the source, also called root. We assume that s is large enough to arrive at all the nodes of the network. In this case, the total stream multicast delay τ_{Ms} for MSM-s is $\tau_{Ms} = (M-1) \cdot s \cdot \mu_r + \tau_s$. That is, the root sends the first message s times and then, $s \cdot \mu_r$ time units later, it begins with the second and so forth. At moment $(M-1) \cdot s \cdot \mu_r$ the root finishes to send the $(M-1)$th message and it begins with the last message, that will arrive at the last node τ_s time units later. Remember that, as shown in Section 2.2, under certain restrictions, the delay τ_s for the last message is the same as the delay for any other message.

Equation for τ_{Ms} is only valid when the root sends each message s times. When s is large the message may be received by all nodes before the root has sent it s times. Though in this case the node could remain idle and wait until $s \cdot \mu_r$ and then begin to send the second message, this would mean a loss of efficiency. So, for MSM-s, when the message is received for all nodes before the root has sent it s times, we will allow the root to send the second message immediately, without an interval of silence. In this particular case the parameter s should be replaced by the actual number of times $s_r(s) \leq s$ that the root sends each message for MSM-s, and then $\tau_{Ms} = (M-1) \cdot s_r(s) \cdot \mu_r + \tau_s$.

Proposition 2 *Given the algorithm MSM-s for $\mathcal{EMPS}$, the delay of a single message is such that $\tau_{\sigma+\Delta} \leq \tau_\sigma \ \forall \sigma, \Delta > 0$.*

Proof: By construction of the algorithm. When bounding up to $\sigma + \Delta$ the transmissions of each node, MSM-($\sigma+\Delta$) will depict a better multicast tree than MSM-($\sigma+\Delta-1$) for any message only if there exists a better solution. Otherwise MSM-($\sigma + \Delta$) will depict the same multicast tree depicted by MSM-($\sigma + \Delta - 1$). Thus $\tau_{\sigma+\Delta} \leq \tau_{\sigma+\Delta-1}$. Repeating the argument for $\tau_{\sigma+\Delta-1}$ and $\tau_{\sigma+\Delta-2}$ and so forth, we obtain $\tau_{\sigma+\Delta} \leq \tau_\sigma \ \forall \sigma, \Delta > 0$. □

Theorem 3 *Given the algorithm MSM-s for $\mathcal{EMPS}(n, \boldsymbol{\lambda}, \boldsymbol{\mu})$, we may obtain the conditions such that MSM-σ is faster than MSM-$(\sigma + 1)$.*

Proof: First we define, in the case that MSM-σ could be better than MSM-$(\sigma + 1)$, the minimum number M_σ of messages from which MSM-σ is better than MSM-$(\sigma + 1)$. We begin with $\sigma = 1$. The value of M_1 can be easily obtained once we have computed τ_1, τ_2 and $s_r(2)$ by executing MSM-1 and MSM-2. Remember that $s_r(2)$ is the number of times that the root sends each message in MSM-2 and that, by Proposition 2, $\tau_1 \geq \tau_2$.

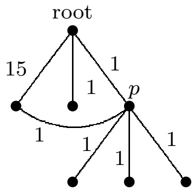


Figure 5: Example of network where $s_r(\sigma) \geq s_r(\sigma+1)$. In particular we have $s_r(3) = 3$ and $s_r(4) = 2$.

Then $\tau_{M1} = (M-1) \cdot \mu_r + \tau_1$ and $\tau_{M2} = (M-1) \cdot s_r(2) \cdot \mu_r + \tau_2$. In this case $s_r(2)$ may be equal to either 1 or 2. In the unusual first case where $s_r(2) = 1$, since $\tau_1 \geq \tau_2$, MSM-2 will be equal or better than MSM-1 for any number of messages. In the more usual case where $s_r(2) = 2$ we establish the restriction $\tau_{M1} \leq \tau_{M2}$ and then $M \geq (\tau_1 - \tau_2)/\mu_r + 1 = M_1$.

For a general case, the number M_σ of messages from which the total stream multicast delay is better for $s = \sigma$ than for $s = \sigma+1$ may be obtained repeating the arguments for M_1. First we obtain by implementing MSM-σ and MSM-$(\sigma+1)$ the following expressions $\tau_{M\sigma} = (M-1) \cdot s_r(\sigma) \cdot \mu_r + \tau_\sigma$ and $\tau_{M\sigma+1} = (M-1) \cdot s_r(\sigma+1) \cdot \mu_r + \tau_{\sigma+1}$. By Proposition 2 we have $\tau_\sigma \geq \tau_{\sigma+1}$. Thus, in the unusual case that $s_r(\sigma) \geq s_r(\sigma+1)$, MSM-$(\sigma+1)$ will be equal or better than MSM-σ for any number of messages. In other case, when $s_r(\sigma) < s_r(\sigma+1)$, we establish $\tau_{M\sigma} \leq \tau_{M(\sigma+1)}$ and then we obtain $M \geq (\tau_\sigma - \tau_{\sigma+1})/((s_r(\sigma+1) - s_r(\sigma)) \cdot \mu_r) + 1 = M_\sigma$. □

Though it is not a usual case, in Figure 5 we depict a network where $s_r(\sigma)$ is greater than $s_r(\sigma+1)$. In particular we have $s_r(3) > s_r(4)$, and thus MSM-4 will be faster than MSM-3 for any number of messages. In order to accomplish the restrictions discussed in section 2.2, we suppose $\mu_r = 2$ and $\mu_p = 1$.

3.2 Analytical Bounds for M_1

In this section, we obtain an analytical bound for M_1, that is, for the number of messages from which the total stream multicast delay is better for $s = 1$ than for $s = 2$. As explained in former section we assume $s_r(2) = 2$. In other case, when $s_r(2) = 1$, MSM-2 will be equal or faster than MSM-1 for any number of messages. Let M be the number of messages; τ_{M1} and

τ_{M2} the multicast delay for MSM-1 and MSM-2 respectively; μ_r the transmission time of the root; and λ_{min} and λ_{max} the minimum and maximum latency between any pair of nodes, respectively. First, we find an upper bound for τ_{M1} and a lower bound for τ_{M2} which we denote respectively by T_1 and t_2. If we force T_1 to be lower or equal than t_2, then MSM-1 will be better than MSM-2:

$$\tau_{M1} \leq T_1 \leq t_2 \leq \tau_{M2} \tag{1}$$

To find T_1 and t_2, we modify slightly the MSM-s performance. First we have:

$$\tau_{M1} \leq (M-1) \cdot \mu_r + (n-1)\lambda_{max} = T_1 \tag{2}$$

Remember that for MSM-1 each node sends each message only once, so MSM-1 depicts a linear tree with $n-1$ links. In this case it is clear that $\tau_{M1} \leq T_1$ since Equation 2 corresponds to the worst case where a message has to cross the $n-1$ links with the maximum latency λ_{max}.

To find a lower bound for τ_{M2} we consider an algorithm with a lower delay than MSM-2. First we assume that the latency for any pair of nodes is the minimum latency λ_{min}. Moreover, in the new algorithm we consider that a node can send the same message simultaneously to two different nodes, that is, that μ_p is equal to 0 for all the nodes. Note that, though this is physically impossible, the new multicast tree will be faster than the tree obtained with MSM-2. Let $N(t), t \in \mathbb{Z}^+$, be the number of nodes that have received the message at step t according to the new algorithm, then $N(t) = 1 + 2 + 4 + \cdots + 2^t = 2^{t+1} - 1$. If we equal $N(t)$ to the number n of nodes we will obtain the number of steps that we need to arrive at all the network $t = \lceil \log_2(n+1) - 1 \rceil$. In this case the new algorithm could send the single message to all the other nodes in $\lceil \log_2(n+1) - 1 \rceil \cdot \lambda_{min}$ time units and then for all the messages we have:

$$\tau_{M2} \geq (M-1) \cdot 2 \cdot \mu_r + \lceil \log_2(n+1) - 1 \rceil \lambda_{min} = t_2 \tag{3}$$

Finally, if according to Equation 1 we force T_1 to be lower or equal than t_2 then τ_{M1} will be also lower or equal than τ_{M2} and MSM-1 will be better than MSM-2. From Equation 2 and Equation 3 it results $M \geq ((n-1)\lambda_{max} - \lceil \log_2(n+1) - 1 \rceil \lambda_{min})/\mu_r + 1$. And since we have considered tighter cases than MSM-1 and MSM-2:

$$M_1 \leq \frac{(n-1)\lambda_{max} - \lceil \log_2(n+1) - 1 \rceil \lambda_{min}}{\mu_r} + 1 \tag{4}$$

Note than when $s_r(2) = 2$ there is always a number of messages from which MSM-1 is better than MSM-2. From Equation 4 we see that this minimum number of messages is linear respect to the number n of nodes. So we can conclude that for the general case that $s_r(2) = 2$, MSM-1 is in general better than MSM-2, provided that the number of messages is usually larger than the number of nodes.

The bound obtained in Equation 4 can be improved by recalculating t_2, that is, by comparing MSM-2 to a tighter algorithm and by using the same lower bound T_1 for MSM-1. We assume that $s_r(2) = 2$. First, we define an algorithm such that, at every step, each node sends the message to one node and such that each node can send the message only twice. We do not consider by the moment time delays. We call $N(t)$ the number of nodes which have received the message at step t. Note that from step $t-1$ to next step t, only the $N(t-1) - N(t-3)$ nodes which have not yet forwarded the message twice can forward it. Thus we have, at step t, the $N(t-1)$ nodes of the last step plus the $N(t-1) - N(t-3)$ nodes that have just received the message one or two iterations before:

$$N(t) = N(t-1) + (N(t-1) - N(t-3)) = 2N(t-1) - N(t-3) \tag{5}$$

In our case we have also $N(0) = 1$, $N(1) = 2$ and $N(2) = 4$. From Table 1 we see that $N(t) = F(t+3) - 1$ where $F(t)$ is the well known Fibonacci serie for $F(0) = 0$ and $F(1) = 1$. Hence, considering $\phi_1 = (1+\sqrt{5})/2)$ and $\phi_2 = (1-\sqrt{5})/2)$ we have:

$$N(t) = F(t+3) - 1 = (\phi_1^{t+3} - \phi_2^{t+3})/\sqrt{5} - 1 \tag{6}$$

As we are determining a lower bound, in order to calculate the number t of steps as a function of the number n of nodes we define $N'(t)$ which is a little faster than $N(t)$ as $N'(t) = (\phi_1^{t+3} + 1)/\sqrt{5} - 1$. Observe that from Equation 6 we have $-1 < \phi_2 < 0$ and then $N'(t) > N(t)$. Hence, if we calculate for $N'(t)$ the number of steps necessary to visit n nodes, we will obtain a lower bound for $N(t)$. For $t \gg 1$ the term ϕ_2^{t+3} is close to 0 and then we have a very accurate bound. If we equal $N'(t)$ to n we obtain $t = \lceil \log_{\phi_1}((n+1)\sqrt{5} - 1) - 3 \rceil$. We can also prove from Figure 6 that at each step t we have a minimum delay of $t \cdot (\lambda_{min} + \mu_{min})/2$ and then:

t				0	1	2	3	4	5	6
N(t)				1	2	4	7	12	20	33

F(t)	0	1	1	2	3	5	8	13	21	34
t	0	1	2	3	4	5	6	7	8	9

Table 1: N(t) vs. Fibonacci Serie.

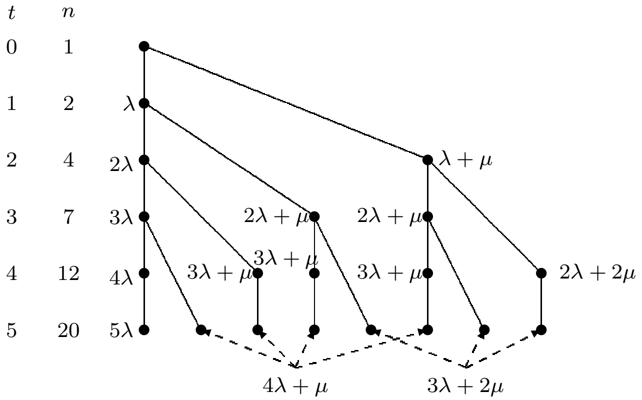


Figure 6: The Fibonacci Tree. Each node forwards the message twice.

$$\tau_{M2} \geq (M-1)\cdot 2\cdot \mu_r + \left\lceil \log_{\phi_1}((n+1)\sqrt{5}-1)-3 \right\rceil \frac{\lambda_{min}+\mu_{min}}{2} = t_2$$

Hence, considering the new bound of t_2 with $\phi_1 = (1+\sqrt{5})/2)$ and repeating the arguments from the former section with the same value of T_1, we have:

$$M_1 \leq \frac{(n-1)\lambda_{max} - \left\lceil \log_{\phi_1}((n+1)\sqrt{5}-1)-3 \right\rceil \frac{\lambda_{min}+\mu_{min}}{2}}{\mu_r} + 1 \qquad (7)$$

This bound is tighter than the bound of Equation 4 depending on the value of μ_{min}. Actually, if μ_{min} is close to λ_{min} the new bound is better than the former, whereas if $\mu_{min} \ll \lambda_{min}$ then we must consider Equation 4. In a practical case we can calculate both bounds and consider the best one.

3.3 An Upper and a Lower Bound for Time Delay in MSM-s

Let τ_{Ms} be the multicast delay for MSM-s and T_s an upper bound of τ_{Ms}. We obtain first a bound for $s = 2$ and then we generalize the result. With this purpose we consider again the algorithm $N(t)$. As we determine an upper bound, in order to calculate the number t of steps as a function of the number n of nodes we define $N''(t)$, which is a little slower than $N(t)$, as $N''(t) = (\phi_1^{t+3} - 1)/\sqrt{5} - 1$. From Equation 6 we have $-1 < \phi_2 < 0$ and then $N''(t) < N(t)$. If we calculate for $N''(t)$ the number of steps that we need to visit n nodes we will obtain an upper bound for $N(t)$. If we equal $N''(t)$ to n we obtain $t = \lceil (\log_{\phi_1}((n+1)\sqrt{5}+1)) - 3 \rceil$. Considering from Figure 6 that at each step t the maximum delay is $t \cdot \lambda_{max}$, we obtain:

$$\tau_2 \leq \lceil (\log_{\phi_1}((n+1)\sqrt{5}+1)) - 3 \rceil \lambda_{max} \tag{8}$$

Note that, if we want to guarantee that $N''(t) < N(t)$, we have to suppose that each node cand send the message twice for MSM-2. Remember also that in section 2.2 we have seen that this is not always possible (in order to avoid congestion). However, we now assume that for MSM-s every node can send the message s times (twice for $s = 2$), even if its rate is lower than the root rate, and then we force the root to send any message with a lower rate than the slowliest node in the graph. In this case, we will not have congestion problems, as referred in section 2.2, and all the messages will have the same delay. Then, since $\tau_s \leq \tau_2 \ \ \forall s \geq 2$ and the root will begin to send a message at most $s \cdot \mu_s$ time units later than the previous one (being μ_s the maximum transmission time of any node in the graph), we obtain:

$$\tau_{Ms} \leq (M-1) \cdot s \cdot \mu_s + \left\lceil \log_{\phi_1}((n+1)\sqrt{5}+1) - 3 \right\rceil \lambda_{max} = T_s \ \ \forall s \geq 2 \tag{9}$$

To find a lower bound t_s for τ_{Ms} we consider, as we did for MSM-2 in section 3.2 but now in a general case, an algorithm such that a node can forward the same message simultaneously to s different nodes. Moreover, we assume that the latency for any pair of nodes is the minimum latency λ_{min}. The new algorithm is therefore better than MSM-s. Let $N(t), t \in \mathbb{Z}^+$, be the number of nodes that have received the message at step t, then $N(0) = 1$ and $N(t) = 1 + s + s^2 + \cdots + s^t = (s^{t+1} - 1)/(s-1)$ for $t > 0$. If we equal $N(t)$ to the number of nodes n we obtain that the number of steps that we need to arrive at all the network for $s \geq 2$ is $t = \lceil (\log_s(n(s-1)+1)) - 1 \rceil$. And thus for $s \geq 2$ we obtain t_s:

$$\begin{aligned} \tau_{Ms} &\geq (M-1) \cdot s_r(s) \cdot \mu_r + \left\lceil \log_s(n(s-1)+1) - 1 \right\rceil \lambda_{min} \\ &\geq (M-1) \cdot \mu_r + \left\lceil \log_s(n(s-1)+1) - 1 \right\rceil \lambda_{min} \end{aligned} \tag{10}$$

For $s = 2$ we obtain the expression in Equation 3 and for $s = 1$ we have $\tau_{M1} \geq (M-1) \cdot \mu_r + (n-1)\lambda_{min}$.

3.4 A General Bound for M_σ

Taking the bounds of the former section and repeating the arguments of section 3.2 for M_1, we obtain a bound for the minimum number M_σ of messages from which MSM-σ is better than MSM-$(\sigma+1)$. As in former sections, this bound has only sense when $s_r(\sigma) < s_r(\sigma+1)$. In other case, MSM-$(\sigma+1)$ is always equal or better than MSM-σ. From Equation 9 we have an upper bound T_σ for $s = \sigma$ and from Equation 10 we obtain a lower bound $t_{\sigma+1}$ for $s = \sigma+1$. Forcing $T_\sigma \leq t_{\sigma+1}$ we will have $\tau_{M\sigma} \leq T_\sigma \leq t_{\sigma+1} \leq \tau_{M(\sigma+1)}$ and then MSM-σ will be better than MSM-$(\sigma+1)$. This results in the next bound for $\sigma \geq 2$:

$$M \geq \frac{\left\lceil \log_{\phi_1}((n+1)\sqrt{5}+1) - 3 \right\rceil \lambda_{max} - \left\lceil \log_{(\sigma+1)}(n\sigma+1) - 1 \right\rceil \lambda_{min}}{s_r(\sigma+1) \cdot \mu_r - \sigma \cdot \mu_s} + 1$$

Note that we assume $s_r(\sigma+1) \cdot \mu_r \geq \sigma \cdot \mu_s$ and thus $s_r(\sigma+1) = \sigma+1$ (since $\mu_r \leq \mu_s$). Otherwise, we should find tighter values for T_σ and $t_{\sigma+1}$ and then recalculate the bound for M. Finally, since we have a pessimistic case, we obtain for $\sigma \geq 2$:

$$M_\sigma \leq \frac{\left\lceil \log_{\phi_1}((n+1)\sqrt{5}+1) - 3 \right\rceil \lambda_{max} - \left\lceil \log_{(\sigma+1)}(n\sigma+1) - 1 \right\rceil \lambda_{min}}{(\sigma+1) \cdot \mu_r - \sigma \cdot \mu_s} + 1$$

3.5 Robustness of MSM-s

Frequently, real-time applications use unreliable transport-layer protocols such as User Datagram Protocol (UDP). That means that it is not always

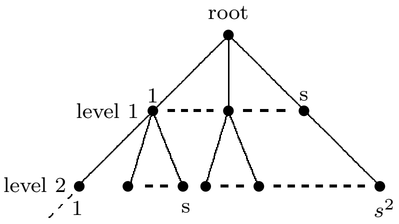


Figure 7: Multicast tree for the calculation of MSM-s robustness.

possible to ensure the ordered and complete arrival of the data at the destination peers. The overlay links of application-layer multicast for real-time applications could therefore provide some degree of reliability. We analyze in this section, under the assumption that there is no message retransmission, the robustness of MSM-s algorithm.

First note that MSM-1 algorithm depicts a linear topology for the multicast tree, that is, a message arrives at a peer which immediately forwards it to another peer and so forth, whereas the MSM-s algorithm depicts in general s divergent paths from each peer of the multicast tree, as shown in Figure 7. In this case, for MSM-1 the probability that a message arrives at l peers is always lower than the probability of arriving at $l-1$ peers. This is because for arriving at the lth peer the message will have to arrive first until the $(l-1)$th peer and then cross successfully the edge between them. This undesirable characteristic does not appears in MSM-s when $s > 1$ due to the different multicast tree that depicts the algorithm, with divergent paths. In MSM-3, for example, the probability of arriving at three peers is much higher than the probability of arriving at only one, an issue which does not happen in MSM-1. Then, as we show in this section, MSM-1 will be less robust than the rest of MSM-s algorithms.

Let $P_c(p,q)$ be the probability that peer q receives correctly a message sent by peer p. For the sake of simplicity we consider that $P_c(p,q) = P_c$ for all the peers. Actually, if we consider $P_c = \max\{P_c(p,q)\}$ $\forall p,q \in \mathcal{EMPS}(n, \boldsymbol{\lambda}, \boldsymbol{\mu})$, we will get a lower bound of the robustness of the MSM-s algorithm. We call a peer which has received correctly the message a "visited peer". Note that since the results would be the same, we calculate the robustness only for the transmission of one single message.

We denote by $\bar{n}_s$ the average number of peers that receive the message for MSM-s with a probability P_c for each peer-to-peer communication. To calculate $\bar{n}_s$ we divide the peers into levels, according to Figure 7. We call E_l the average number of peers that receive the message at level l. For the first level we have s peers and then:

$$E_1 = E(r_{11} + r_{21} + \cdots + r_{s1}) = sE(r_{11}) = s(0 \cdot p(0) + 1 \cdot p(1)) = sP_c$$

By definition r_{ij} is 1 if the peer i at level j has received the message and 0 otherwise (we calculate the average number of visited peers when we send only one message). Thus $r_{11} + r_{21} + \cdots + r_{s1}$ is equal to the number of peers that have received the message at the first level. For the s^2 peers of the second level we have:

$$E_2 = E(r_{12} + r_{22} + \cdots + r_{s^2 2}) = s^2 E(r_{12}) = s^2(0 \cdot p(0) + 1 \cdot p(1)) = s^2 P_c^2$$

And in general for the level l:

$$E_l = E(r_{1l} + r_{2l} + \cdots + r_{s^l l}) = s^l E(r_{1l}) = s^l(0 \cdot p(0) + 1 \cdot p(1)) = s^l P_c^l$$

Finally we calculate $\bar{n}_s$ as the sum of the averages of each level, considering that, since the root always has the message, for level 0 this number is 1. We denote by L the number of levels:

$$\begin{aligned} \bar{n}_s &= E_0 + E_1 + \cdots + E_L \\ &= 1 + sP_c + (sP_c)^2 + \cdots + (sP_c)^L \\ &= \frac{(sP_c)^{L+1} - 1}{sP_c - 1} \end{aligned} \tag{11}$$

In this case we assume that the number of peers is $1 + s + s^2 + \cdots + s^L = (s^{L+1} - 1)/(s - 1)$ and that we flood the network level by level (this only would happen on a very regular network with little time transmissions). For the MSM-1 algorithm the assumptions of Equation 11 are valid. Since we have $L = n - 1$ it results:

$$\bar{n}_1 = \frac{1 - P_c^n}{1 - P_c} < \frac{1}{1 - P_c}$$

Thus if $1/(1 - P_c)$ is much smaller than the number n of peers, the average number of visited peers with MSM-1 will be also much smaller than n. But if, on the contrary, $1/(1 - P_c)$ is higher than n then the average number of visited peers may be close to n.

n	$P_c = 0.9$	$P_c = 0.99$	$P_c = 0.999$	$P_c = 0.9999$
50	9.95	39.50	48.79	49.88
100	10.00	63.40	95.21	99.51
1000	10.00	100.00	632.30	951.67

Table 2: Average of the number of peers that receive the message for MSM-1, depending on n and P_c.

In Table 2 we see some values of the average for MSM-1 depending on n and P_c. For $P_c = 0.9$ we have $1/(1 - P_c) = 10$ and then the average may not be greater than 10, no matter how high is n. However, for the usual values of $P_c = 0.999$ and $n = 50$ or $n = 100$ we have good averages, close to n. For the other values the average is much smaller than n, which means that the message is not received by a large percentage of peers. In this case the algorithm should automatically change from MSM-1 to MSM-2. For instance, for $n \approx 1000$ and $P_c = 0.999$ we would arrive at only the 63.2% of the peers with MSM-1 whereas for MSM-2 the percentage would be of 99.1%. In this case the percentage for MSM-3 would be only a little higher than for MSM-2: 99.4%. Actually, in a general case the robustness of MSM-2 will be acceptable.

Therefore, when we want to apply MSM-1 to a real network (assuming that MSM-1 can topologically arrive at all the peers with the restriction $s = 1$, that the rate of the source is high enough to provide a new message every μ_r time units, and that for the number of messages that we have the time delay is lower for MSM-1 than for MSM-2), in this case the algorithm itself should estimate the average number of peers that will receive the message for MSM-1 and if it would not be high enough then it should apply MSM-2, consider the new average number and change if necessary to MSM-3, and so forth.

Nevertheless, the assumption that P_c is equal for all the links has major implications on MSM-1 than on MSM-s for $s \geq 2$. In general, in MSM-1 there are a larger percentage of short end-to-end transmissions than in MSM-s for $s \geq 2$. This means that in MSM-1 there will be in general more transmissions than in MSM-s with a probability of success larger than P_c. Thus, the results of Equation 11 can be more pessimistic for $s = 1$ than for $s \geq 2$.

4 Conclusions and Future Work

In this paper we propose an algorithm, called MSM-s, for multicast transmissions in application-layer networks. The fundamental parameter of MSM-s is the value of s, i.e. the maximum number of times that a peer may forward a single message. This parameter is also the maximum degree of each node in the resulting multicast tree. We present a theoretical study of the multicast delay and the robustness of the algorithm.

The practical implementation of the algorithm in real peer-to-peer networks will be part of our future work. Though in this paper we have not considered the possibility of dynamic multicast groups, we plan to define mechanisms for the actualization of the routing tables which take into account the joining and leaving of peers without recalculating the whole table. This could mean a loss of efficiency but would simplify the computation. We consider the possibility of allowing a maximum number of joinings and leavings of peers with only partial changes on routing tables. Once this number is achieved the algorithm would proceed to completely recalculate the tables. This maximum number could be determined by means of theoretical bounds and network measures.

We also plan to study the benefits of the use of the algorithm for two real-time applications: multi-player networking games, which can be considered a static scenario with large restrictions on delay and with multiple points of information, and video-streaming of stored content and live-events, which are dynamic scenarios where the preservation of messages rate is of great importance.

References

[1] A. Bar-Noy and S. Kipnis. Designing broadcasting algorithms in the postal model for message-passing systems. *Proceedings of the Fourth Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA '92*, 13–22, 1992.

[2] M. Castro, P. Druschel, A-M. Kermarrec and A. Rowstron. SCRIBE: A large-scale and decentralised application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communication*, 20(8):85–110, 2002.

[3] Y. Chawathe, S. McCanne and E. A. Brewer. RMX: Reliable Multicast for Heterogeneous Networks. *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2000*, 2:795–804, 2000.

[4] S. E. Deering and D. R. Cheriton. Multicast routing in datagram internetworks and extended LANs. *ACM Trans. Comput. Systems*, 8(2):85–110, 1990.

[5] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek and J. W. O'Toole. Overcast: Reliable Multicasting with an Overlay Network. *USENIX Symposium on Operating Systems Design and Implementation*, 197–212, 2000.

[6] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Shenker. A Scalable Content Addressable Network. *Proceedings of the ACM SIGCOMM 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 161-172, 2001.

[7] J. H. Saltzer, D. P. Reed and D. D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277–278, 1984.

[8] E. W. Zegura, K. L. Calvert and S. Bhattacharjee. How to Model an Internetwork. *Proceedings of the Fifteenth Annual Joint Conference of the IEEE Computer Societies, Networking the Next Generation, INFOCOM '96*, 2:594–602, 1996.

[9] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz and J. D. Kubiatowicz. Bayeux: An Architecture for Scalable and Fault-tolerant Wide-area Data Dissemination. *Proceedings of the 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video, NOSSDAV '01*, 11–20, 2001.